02

Funções que trabalham com números

Transcrição

Nós já vimos as duas funções famosas: `round()`, usada para arredondamentos, e `trunc()`, para "truncar".

Com a `ROUND`, os critérios de arredondamento eram: de 0 até 4, o arredondamento era feito para menos; de 5 até 9, o arredondamento era para mais. Temos as duas funções de arredondar para mais ou menos, separadas no Oracle.

Vamos trabalhar com o valor de π (pi) igual a 3,14. Para arredondá-lo para cima, temos uma função chamada `ceil()`, que trabalha com a ideia de "forrar" o número.

```
SQL> select ceil(3.14) from dual;

CEIL(3.14)
----------
         4
```

Porém, podemos querer arredondar a partir de duas casas decimais. Se tentarmos usar o valor `2`, como segundo parâmetro, ele irá responder que o número de argumentos é inválido. Porque ele sempre irá aumentar o número inteiro.

```
SQL> select ceil(3.14,2) from dual;

CEIL(3.14)
----------
         4

SQL> select ceil(3.14,2) from dual;
select ceil(3.14,2) from dual

ERRO na linha 1:
ORA-00909:numero de argumentos invalido
```

Isto significa que não conseguimos especificar a quantidade de casa decimais que desejamos trabalhar. Ele sempre irá retorna o número maior ao valor inteiro, que estou usando como parâmetro.

Podemos também testar o que acontece se usamos um valor nulo.

```
SQL> select ceil(null) from dual;

CEIL(NULL)
----------
```

E o que acontece se informamos um valor negativo?

```
SQL> select ceil(-3,14) from dual;

CEIL(-3,14)
```

```
-----------
         -3
```

O maior valor inteiro será `-3`. Vale ficar atento ao valor de arredondamento quando o número for negativo.

Temos também uma função que arredonda para menos: `floor()`.

```
SQL> select floor(3,14) from dual;

FLOOR(3,14)
-----------
         3
```

O valor `3,14` arredondado para menos, será igual a `3`. Mesmo quando o número decimal estiver mais próximo de `3`, ele irá reduzir o valor.

```
SQL> select floor(2.9) from dual;

FLOOR(2.9)
-----------
         2
```

Além das duas funções, temos uma que calcula a raiz quadrada - um número que multiplicado por ele mesmo, resulta em um determinado valor.

Se calcularmos a raiz quadrada de `81`, o resultado será `9`. Usaremos a função que calcula a *square root* (que significa "raiz quadrada", em inglês): `sqrt()`.

```
SQL> select sqrt(81) from dual;

  SQRT(81)
----------
         9
```

Se calcularmos a raíz de `0`, o resultado será o mesmo número:

```
SQL> select sqrt(0) from dual;

  SQRT(0)
----------
         0
```

Porém, na matemática não podemos calculara a raiz quadrada de números negativos. Por isso, se usarmos a função, o resultado será negativo.

```
SQL> select sqrt(-5) from dual;
select sqrt(-5) from dual
```

```
ERRO na linha 1:
ORA-01428: o argumento '-5' esta fora da faixa valida
```

E se passarmos um valor `null`, o resultado será nulo.

```
SQL> select sqrt(null) from dual;

SQRT(NULL)
----------
```

Além da raiz quadrada outra operação comum é descobrirmos a potência. Podemos dizer que potenciação representa uma multiplicação de fatores iguais. Por exemplo, 2^3, que representa `2*2*2`. Ou seja, o número `2` multiplicado três vezes por ele mesmo. Uma maneira de calcularmos isto é com a função `power()`. Vamos testar usando a base `2`, elevado à potência `3`.

```
SQL> select power(2,3) from dual;

POWER(2,3)
----------
         8
```

Ele irá retornar o valor `8`.

Se quiser saber quanto é `2` elevado a décima potência, podemos especificá-la no segundo parâmetro.

```
SQL> select power(2,10) from dual;

POWER(2,10)
-----------
        1024
```

Se usarmos a base igual a `0`, ele irá retornar `0`.

```
SQL> select power(0,10) from dual;

POWER(0,10)
-----------
          0
```

Faz sentindo, considerando que `0` multiplicado várias vezes por ele mesmo, resultará em `0`.

E o que acontece se passarmos um número negativo? Por exemplo, `-5` elevado à décima potência.

```
SQL> select power(-5,10) from dual;

POWER(-5,10)
------------
     9765625
```

Podemos testar com o segundo parâmetro com um valor negativo.

```
SQL> select power(-5,10) from dual;

POWER(2,-10)
------------
  ,000976563
```

Também, existe uma maneira de isolarmos o expoente da potência. Isto significa que podemos perguntar, considerando que temos a base `2`, quantas vezes precisamos multiplicá-lo por ele mesmo, para resultar em `1024`? Neste caso, queremos calcular um logaritmo: `log()`.

```
SQL> select log(2,1024) from dual;

LOG(2,1024)
-----------
         10
```

O primeiro parâmetro é a base e o segundo é referente ao valor total. O resultado da função será o valor da potência.

E se testarmos com uma base de valor negativo?

```
SQL> select log(-5,9765625) from dual;
select log(-5, 9765625) from dual

ERRO na linha 1:
ORA-01428: o argumento '-5' esta fora da faixa valida
```

Não conseguimos calcular o `log()`, quando o número é negativo. Mas se trabalharmos com o valor `5`, o resultado será diferente.

```
SQL> select log(5,976562) from dual;

LOG(5,976562)
-------------
           10
```

Basicamente, usamos a função `log()` para isolar o segundo parâmetro da função `power()`. Não vamos nos aprofundar sobre os conceitos de **potência** e **algoritmo***, porque não serão cobrados no exame. Mas é importante conhecer as duas funções.

Outra função que precisamos conhecer é a que trabalha com o número de **Euler** (ou **neperiano**) - a base dos logaritmos naturais. Nós podemos usar a função `exp()`, que irá calcular a potência com a base sendo o número de Euler. Vamos testar a função com o valor `1`.

```
SQL> select exp(1) from dual;

    EXP(1)
```

```
----------
2,71828123
```

Ele irá retornar o próprio número de Euler, porque qualquer número elevado a `1`, será ele mesmo. O número `2,71828123` é importante para a matemática, porém no Oracle já teremos as operações para trabalharmos com ele. Se passarmos o número `2` para a função, ele irá calcular o número neperiano elevado a `2`.

```
SQL> select exp(2) from dual;

   EXP(2)
---------
7,3890561
```

Podemos fazer o cálculo contrário também e descobrir o logaritmo na base do número neperiano, com a função `ln()`. Vamos testá-la usando o próprio número neperiano.

```
SQL> select ln(2.71828183) from dual;

LN(2.71828183)
--------------
             1
```

Ele retornou o valor `1`, porque realizamos exatamente a operação inversa da função `exp()`.

Vale lembrar que a prova não irá cobrar o tema sobre o número de Euler, mas é importante conhecer as funções.

Faltou conhecermos uma operação bastante usada, que determina o resto de uma divisão. Para fazermos isto, usaremos a função `mod()` e passaremos como parâmetro `13` dividido por `5`.

```
SQL> select mod(13,5) from dual;

 MOD(13,5)
----------
         3
```

O resultado de `13/5` será `2`, com um resto igual a `3`. Veremos com o `MOD` funciona com a divisão `13/5`.

```
SQL> select mod(16,5) from dual;

MOD(16,5)
---------
        1
```

O resultado de `16/5` será `3`, com um resto igual a `3`.

Porém, temos outra forma de encontrar o resto da divisão é utilizando a função `remainder()`.

```
SQL> select remainder(13,5) from dual;
```

```
REMAINDER(13,5)
--------------
            -2
```

O resultado foi `-2`. Mas ele deveria retornar o resto? O resultado é diferente de quando usamos o `MOD`. Vamos ver o que acontece com a operação

```
SQL> select remainder(16,5) from dual;

REMAINDER(16,5)
---------------
              1
```

Agora, o retorno foi igual a outra função usada. Por que isto está acontecendo? A função `remainder()` é semelhante à `mod`, mas possui uma pequena diferença: o `MOD` usa o `FLOOR` no cálculo do resultado e o `REMAINDER` usa o `ROUND`.

No caso da divisão `13/5`, quando usamos o `REMAINDER`, ele irá retornar `-2`, porque significa que o número `5` cabe `3` vezes no `13`, mas "passará" `2`.

O `MOD` quando mostra quantas vezes um número cabe dentro de outro, ele arredonda para menos, usando o `FLOOR`, enquanto o `REMAINDER` arredonda para mais, usando o `ROUND`.

Estas são as principais funções para trabalharmos com números. Quando eu fiz o exame de certificação, as funções mais cobradas foram: `ROUND`, `MOD`, `REMAINDER`. `TRUNC` e `SIGN`. Vale a pena prestar atenção nestas durante a sua preparação.